The iObserve Approach
--
Runtime Architecture Modeling and Visualization
### Schedule of Events

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00 – 09:10</td>
<td>Welcome and General Introduction</td>
</tr>
<tr>
<td>09:10 – 09:40</td>
<td>Study Foundations</td>
</tr>
<tr>
<td>09:40 – 10:00</td>
<td>Model-based Software Application Monitoring</td>
</tr>
<tr>
<td>10:00 – 10:30</td>
<td><strong>Runtime Architecture Modeling and Visualization</strong></td>
</tr>
<tr>
<td>10:30 – 11:00</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>11:00 – 12:15</td>
<td>Introduction to the ExplorViz, Palladio, and iObserve Approaches with following Tool / Visualization Demos</td>
</tr>
<tr>
<td>12:15 – 12:30</td>
<td>Study Setup</td>
</tr>
<tr>
<td>12:30 – 14:00</td>
<td>Lunch</td>
</tr>
<tr>
<td>14:00 – 15:30</td>
<td>Comprehensibility Study</td>
</tr>
<tr>
<td>15:30 – 16:00</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>16:00 – 16:30</td>
<td>Live Database Trace Visualization in Large Software Landscapes</td>
</tr>
<tr>
<td>16:30 – 17:00</td>
<td>Feedback and Open Discussion</td>
</tr>
</tbody>
</table>
A software application build by composing services

- Flexibility, scalability, reusability
- Economic use of resources
- Complexity, fragility
- Changes during operations unknown in development phase

DevOps practices

R. Heinrich
ICSA 2017 Tutorial Runtime Modeling and Visualization
DevOps

DevOps is a set of practices of operators and developers participating together in the entire application lifecycle, from design through the development process to production support.

E. Mueller. What is DevOps?

- Developers and operators must work more closely
- Feedback cycles from Dev to Ops and Ops to Dev
- Increased communication among developers and operators
- Integration of the role of developer and operator

Differences in architectural models in development and operations

R. Heinrich
ICSA 2017 Tutorial Runtime Modeling and Visualization
Level of abstraction (component-based vs. close to implementation level)
# Differences in Architectural Models in Development and Operations

1. **Level of abstraction** (component-based vs. close to implementation level)

2. **Purpose** (finding appropriate design vs. reflecting current application configuration)

3. **Content** (static vs. dynamic)
   - Structure and design
   - In-memory objects and communication
   - Utilization of server

- Limited reuse of development models during operations
- Limited phase-spanning consideration of the software architecture
Overview of the iObserve Approach

DFG Priority Programme 1593
Design For Future - Managed Software Evolution

R. Heinrich
ICSA 2017 Tutorial Runtime Modeling and Visualization
The iObserve Megamodel

Bridges the divergent abstraction levels in development and operations


R. Heinrich ICSA 2017 Tutorial Runtime Modeling and Visualization
Operator-in-the-Loop Adaptation to involve operator adaptation decisions

Monitor

Analyze

Plan

Execute

Descriptive architectural run-time models

Prescriptive architectural run-time models

Conclusion and Future Work

Differences in architectural models in development and operations
- Abstraction (component-based vs. close to implementation level)
- Purpose (finding appropriate design vs. reflecting executed application)
- Content (static vs. dynamic)

➢ iObserve is a first attempt to bridge differences in architectural models

Future Work
- Further investigate the planning and execution phases
- Improve live visualization of data contained in the iObserve megamodel
- Conduct experiments for evaluating scalability and usefulness

- Develop reference architecture for various quality aspects in monitoring, analysis and planning
- Support meta-model modularity, extension and evolution
https://github.com/research-iobserve
References